Automate Nested Cloud Account Discovery and Integration with Morpheus

Last Updated: December 2020

INTRODUCTION

Morpheus is a powerful hybrid cloud application orchestration platform capable of integrating with numerous public and private cloud technologies to establish a consolidated approach to PlatformOps and developer self-service. Additionally, Morpheus supports many scripting and automation tools to cover all phases of the application lifecycle. This whitepaper will focus on up-front cloud integration, specifically how large organizations with many cloud accounts can ease the burden of discovering new accounts and integrating them into Morpheus.

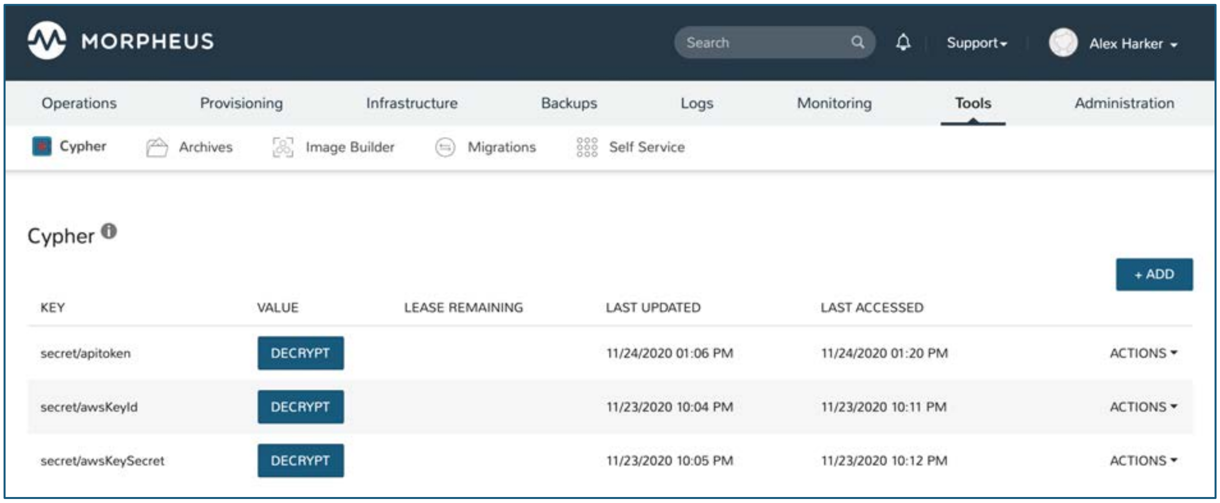
For this example, I have an 'AWS Organization', which is an umbrella account under which new AWS accounts can be created or added. Many customers use this functionality to simplify billing and policy enforcement across a large number of AWS accounts. While we're using AWS here, this same scenario could be represented in Azure using a similar construct such as Management Groups or even with other public clouds. Within my Organization are several subaccounts in addition to the managing account. My goal is to scan my Organization for any accounts added or created within the last day, harvest the relevant credentials, and integrate any newly added AWS accounts as Clouds in Morpheus.

Exactly how you go about this will depend heavily on the clouds you're using and your organization's IAM policies, but this example illustrates the concept that Morpheus is highly flexible and extensible in allowing administrators to deploy highly automated strategies.

We'll be using a Python script in this example along with Morpheus integration with GitHub code repositories, secret storage through Cypher, automation tasks, timed jobs, and the Morpheus API. The outcome is full automation of what would otherwise be a time-consuming and tedious manual chore subject to human error.

1. ENABLE SECRET STORAGE WITH MORPHEUS CYPHER

To get started, I'll make sure the relevant secrets are available through Morpheus Cypher (Tools > Cypher). In my script, I invoke a Key ID and Key Secret to make programmatic calls to AWS and I will also invoke an API token for the Morpheus API. These secrets can be securely called into my script when it's run as a Morpheus Task and no secrets are revealed in the script itself meaning I can freely share my code or upload it to public repositories.

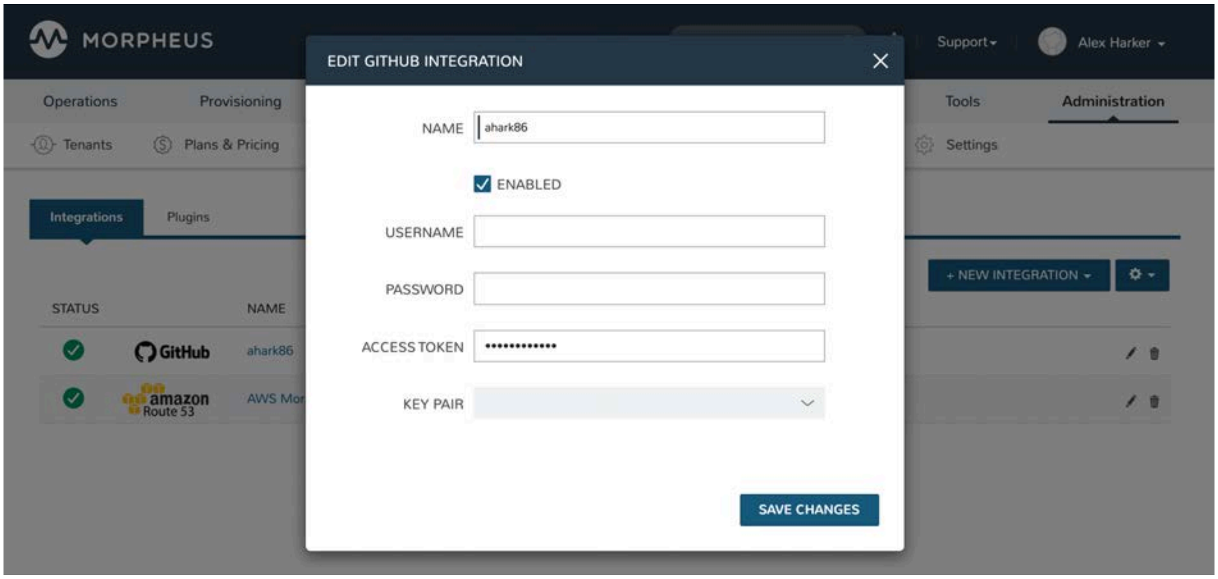


KEY	VALUE	LEASE REMAINING	LAST UPDATED	LAST ACCESSED	
secret/apitoken	DECRYPT		11/24/2020 01:06 PM	11/24/2020 01:20 PM	ACTIONS
secret/awsKeyId	DECRYPT		11/23/2020 10:04 PM	11/23/2020 10:11 PM	ACTIONS
secret/awsKeySecret	DECRYPT		11/23/2020 10:05 PM	11/23/2020 10:12 PM	ACTIONS

2. INTEGRATE A GITHUB REPOSITORY

My Python script is kept under Git version control and uploaded to a public GitHub repository. By integrating my GitHub account with Morpheus, I can ingest any repository as a Morpheus automation task. When code changes are pushed to this repository, Morpheus always executes the most recent version of the code whenever the task is called. New GitHub integrations can be created in Administration > Integrations.

If you wanted to skip the repository integration you could also simply paste the Python script directly into a new automation Task as described in step 3. Simply set the "Source" value to Local on your new Task rather than Repository as I've done in my example to call in the script from the integration source.



NAME aharks86

☑ ENABLED

USERNAME

PASSWORD

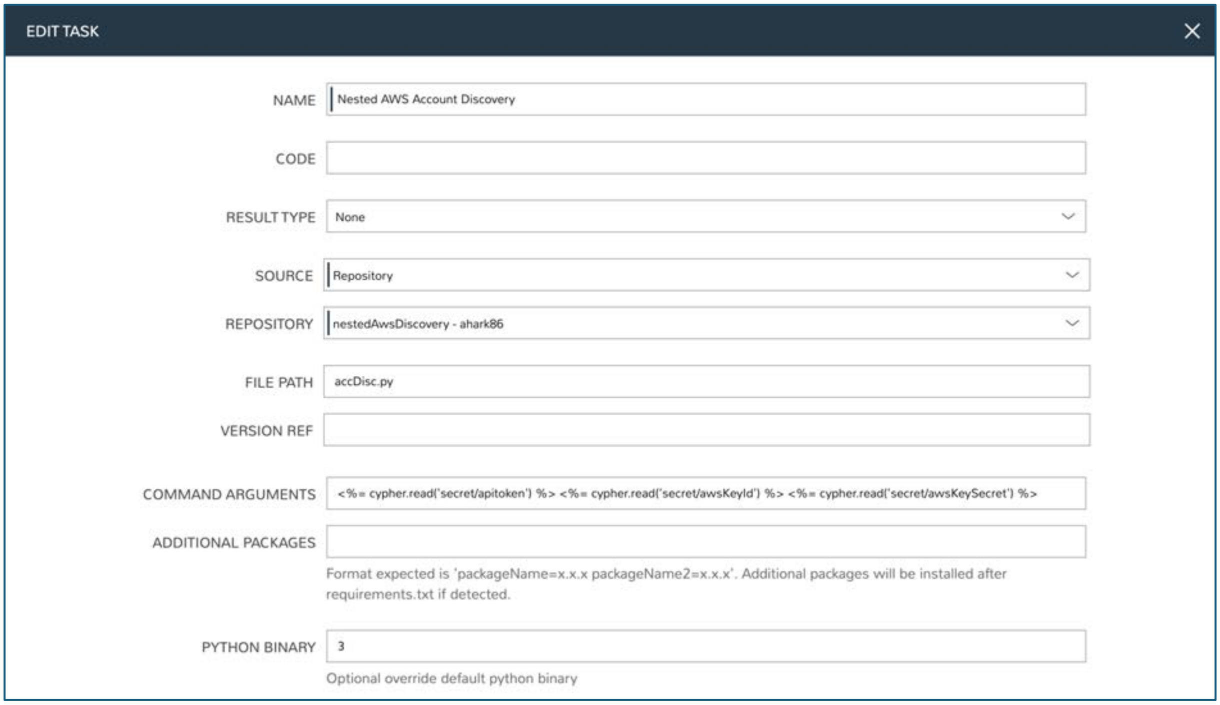
ACCESS TOKEN •••••••••••••

KEY PAIR

3. CREATE A MORPHEUS TASK

Tasks in Morpheus come from many sources and can include scripts of various types, API (HTTP) calls, Ansible Playbooks and more. In this case, we're going to create a Python script Task directly from our GitHub integration.

New Tasks are started from Provisioning > Automation. Click "+ADD" and change the Task Type to a Python Script. In my case, I'm sourcing the script code from a repository, but you can opt to write or paste code directly into this modal. I'm also setting my Cypher secrets as command line arguments as shown in the screen shot below. Once the Task is configured correctly, click "SAVE CHANGES". As mentioned previously, Morpheus will always run the current version of this code. If we push changes to the GitHub repository, we don't need to touch the Task in Morpheus.



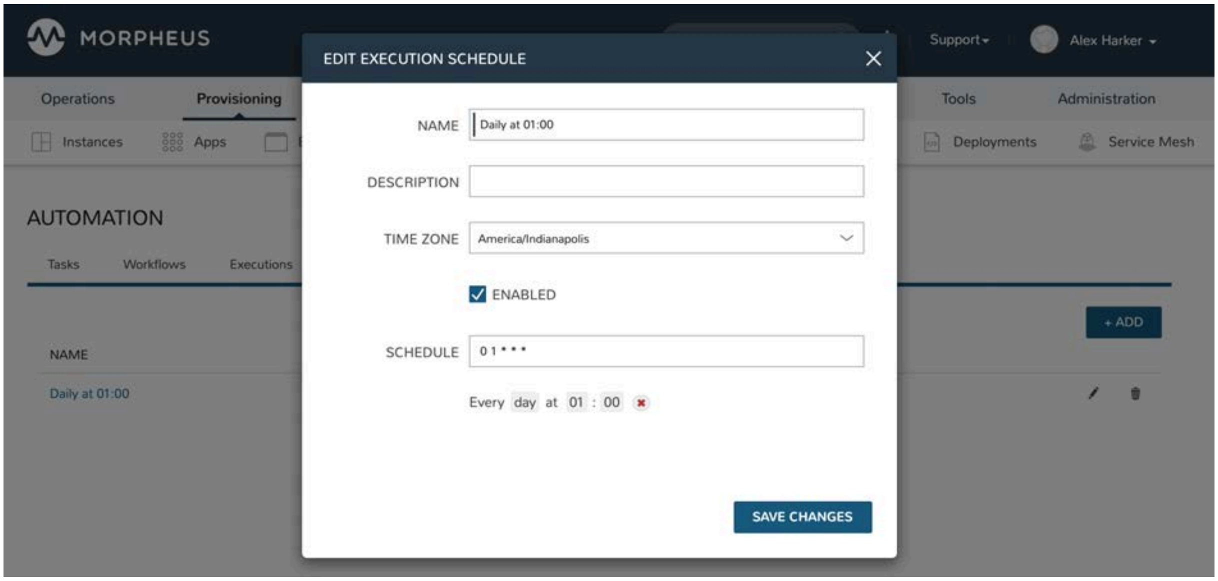
EDIT TASK

NAME	Nested AWS Account Discovery
CODE	
RESULT TYPE	None
SOURCE	Repository
REPOSITORY	nestedAwsDiscovery - ahark86
FILE PATH	accDisc.py
VERSION REF	
COMMAND ARGUMENTS	<%= cypher.read('secret/apitoken') %> <%= cypher.read('secret/awsKeyId') %> <%= cypher.read('secret/awsKeySecret') %>
ADDITIONAL PACKAGES	
	Format expected is 'packageName=x.x.x packageName2=x.x.x'. Additional packages will be installed after requirements.txt if detected.
PYTHON BINARY	3
	Optional override default python binary

4. AUTOMATE THE TASK RUN WITH MORPHEUS JOBS

In this case, I've structured my Python code to check for new AWS accounts associated with my Organization over the previous day. In light of that, I'd like Morpheus to run this Task once per day so new accounts are automatically integrated into Morpheus by the following morning. We can automate Task runs with Morpheus Jobs.

First, check to see if an existing execution schedule meets our needs. Execution schedules are stored in Provisioning > Automation > Execute Scheduling. If needed, click "+ADD" to begin a new schedule. The schedule I'm using for my example is shown below:



EDIT EXECUTION SCHEDULE

NAME: Daily at 01:00

DESCRIPTION:

TIME ZONE: America/Indianapolis

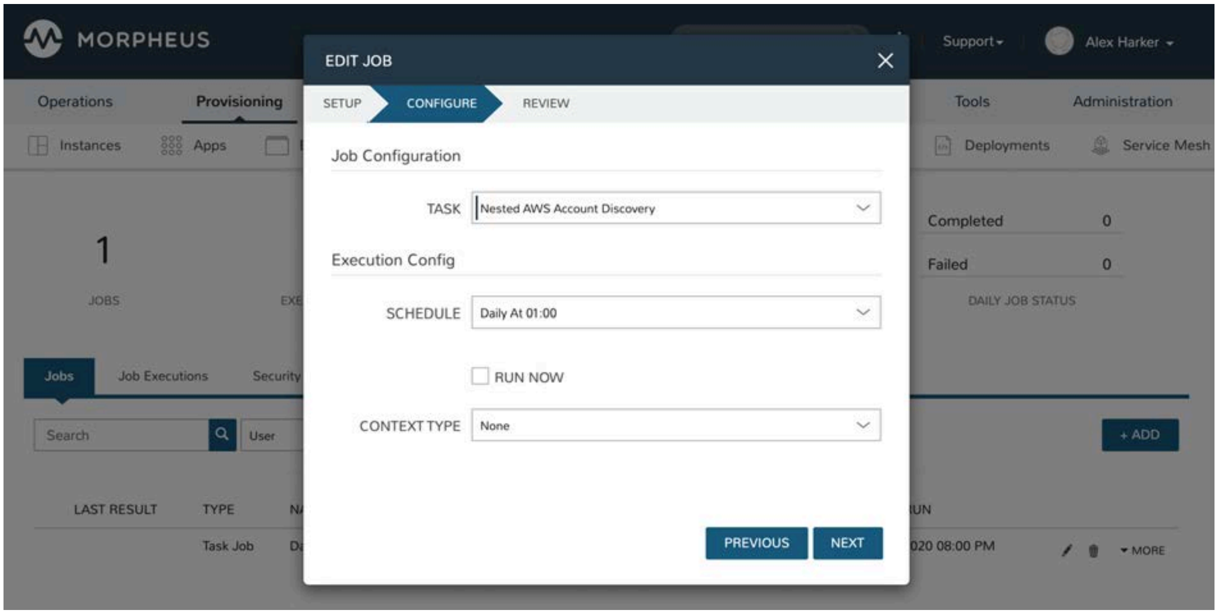
☑ ENABLED

SCHEDULE: 0 1 * * *

Every day at 01 : 00

SAVE CHANGES

With the execution schedule configured, move to Jobs (Provisioning > Jobs). At the time of this writing, Jobs consist of Tasks, Workflows or Security Scans which are run either on-demand or on a selected execution schedule. In this case, we want a scheduled Task Job so our Python Task runs once per day in the early morning hours. We even have the option to run the Job at the time its first saved so we can go ahead and check for any new AWS clouds that were added yesterday before leaving our Job to continue running overnight indefinitely. The configuration for my Job is shown below:



EDIT JOB

SETUP · CONFIGURE · REVIEW

Job Configuration

TASK: Nested AWS Account Discovery

Execution Config

SCHEDULE: Daily At 01:00

□ RUN NOW

CONTEXT TYPE: None

PREVIOUS · NEXT

CONCLUSION

In this example I've used Morpheus tools to automate a daily process of checking for and integrating any new AWS accounts into my Morpheus appliance. By integrating these accounts in an automated fashion, I ensure new accounts are always integrated into my CMP daily without any human effort or potential for error. I also ensure I maintain governance over all cloud accounts to ensure resources are used efficiently and costs are kept in check. Morpheus tools allow me to easily consume code from my online repositories, run it on a scheduled basis, keep track of secret values, and more.

The code used in this example is available here, in our official repository for automation script examples. At the time of this writing, the repository is just getting its start, but you may want to bookmark that page or follow us on GitHub to see additional automation examples from Morpheus engineers. Feel free to adapt this example as needed for your purposes as a starting point for deploying a similar solution in your environment. Bear in mind that even though this code works for my purposes, it is simply an example. Morpheus makes no guarantee this code will work for you or that it would be safe to run in your environment.

If you're ready to get started with Morpheus, deploy our Community Edition in your home lab for free right now! Morpheus Community Edition is time-unlimited and allows you to integrate up to three Clouds and on-board up to 25 workload elements. It's perfect for managing a small set of workloads in a home lab and allows unlimited experimentation with all of the features mentioned in this brief.